

Lab 12

Loops

GOALS:

1. Students will be able to create programs using a loop that repeats code.
2. Students will be able to trace a program that has loops.

ESSENTIAL QUESTIONS:

1. List two ways loop structures are classified.
2. What are the 3 parts to all loops?
3. Can you interchange loops?

OVERVIEW ON LOOPS:

There are three types of loops

- | | | |
|--|--|--------------------------------|
| 1. do
code
Loop while <u>Boolean expression</u> | 2. do while <u>Boolean expression</u>
code
loop | 3. For To Step
code
Next |
|--|--|--------------------------------|

Loops are classified two different ways:

1. When the test occur Pre-Test or Post test
2. By the number of times it loops Fixed or Variable

Three parts to all loops

1. Initialize Variable (priming the loop) 'setting a starting value
2. Terminating condition 'Boolean expression or variable that evaluates to false to stop the loop.
3. Increment/Decrement variable 'Variable that is increased or decrease to make the Boolean variable False.

Do While Loop. Its syntax is the following:

Do While (Expression)
(Code to execute)
Loop

(Expression) can be any legal logical expression that we wish to evaluate to determine whether or not to exit the loop. Each time the program reaches Loop it will verify that this expression is True, and if it is False, it will exit the loop for us. Thus, instead of exiting when an expression is True, it now exits only once this expression is false. Let's try rewriting our Fibonacci program to use a Do-While loop instead of a Do-Until loop.

```
private Sub cmdClick_click()  
    Dim X As Integer  
    Dim Y As Integer  
    Dim cnt As Integer  
    cnt = 1  
    Do While cnt < 8  
        Debug.Print X  
        X = Y + X  
        Y = X - Y  
        cnt = cnt + 1  
    Loop  
End Sub
```

'Our counter.
'1, 1, 2, 3, 5, 8, 13, 21, ...

For-Next Loops

In situations where you merely want to run the loop a *predefined number of times*, it can become quite tiresome to have to create and manage a counter for each loop, which is why we also have something called a For-Next Loop. This kind of loop allows you to specify a counter, to tell it to count from one number to another each time through the loop, and to exit once the counter has reached its upper limit. The syntax is as follow:

```
Dim i As Integer
```

```
For i = (Integer) To (Integer)
    (Code to execute)
Next i
```

We used the variable name "i" above, as it is the most common name used for For-Loops; however, you can use any variable name you want, so long as the variable is of the type Integer. Now, let's improve our Fibonacci program even further:

```
Public Sub Main()
    Dim X As Integer
    Dim Y As Integer
    Dim cnt As Integer 'Our counter.

    For cnt = 1 To 8
        Debug.Print X
        X = Y + X
        Y = X - Y
    Loop
End Sub
```

In the example above, we first dimensioned cnt as an Integer, and then, in the declaration of the For-Next loop, set its value to 1. Each time through the loop, the value of cnt was incremented by 1 until it reached 8, at which point the loop was executed.

Step - (optional)

By default, the variable used in the declaration of the For-Next loop is incremented by 1 each time through the loop; however, if you want to increment this value by a different amount each time through the loop, you can simply append Step (Integer) to the end of the For-Next loop declaration. If, for instance, we wanted to print out every even number counting backward from 20 to 0, we could do this using the following code:

```
Dim I As Integer

For I = 20 To 0 Step -2
    Debug.Print I
Next I
```

The Post-Test Loop

With a post-test loop, the "loop termination decision" is tested at the bottom of the loop, therefore the statements in the body of the loop will always be executed at least one time.

In VB, post-test loops are implemented with the Do/Loop statements, however, the "While" condition appear after the keyword Loop, not Do. The "While" versions of the post-test loop are flowcharted below. The only difference between the two is the placement of the "True" and "False" paths extending from the decision diamond.

The Do/Loop While (Loop Until) Loop

The general format for a post-test loop in VB is:

Do

<list of statements>

Loop While <condition>

As an example of a post-test loop, the following is a code segment containing a Do/Loop construct that allows a user three tries to get his password right:

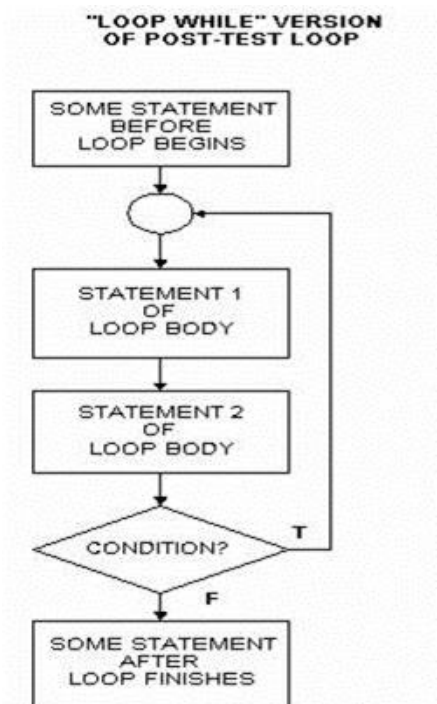
```
Dim strPassWord As String
Dim strUserTry As String
Dim intNumTries As Integer
strPassWord = "BEAVIS"
intNumTries = 0
```

Do

```
strUserTry = InputBox("Enter password: ")
```

```
intNumTries = intNumTries + 1
```

```
Loop while strUserTry = strPassWord Or intNumTries = 3
```



Counters:

Counter is a variable that gets increased or decreased by a constant.

Examples:

```
intX = intX + 1
```

```
intX = intX - 1
```

Accumulators:

Accumulators are similar to counters. An accumulator keeps a running total or sum of what is being **inputted**.

Example:

```
intValue = txtInput.Text
```

```
intSum = intSum + intValue
```

Infinite Loops:

Infinite loops are loops that never stop looping. The Boolean expression/variable never becomes false.

To escape infinite loops press CTRL + BREAK

Labs

Create the following projects

Project 1

1. Create the programs below
2. Save it as frmPrimeNumbers
3. Code the following

Option Explicit

```
Private Sub txtInteger_Change()  
    lblPrimeResult.Caption = ""  
End Sub
```

```
Private Sub cmdTest_Click()  
    Dim intTestNum As Integer  
    Dim intDivisor As Integer
```

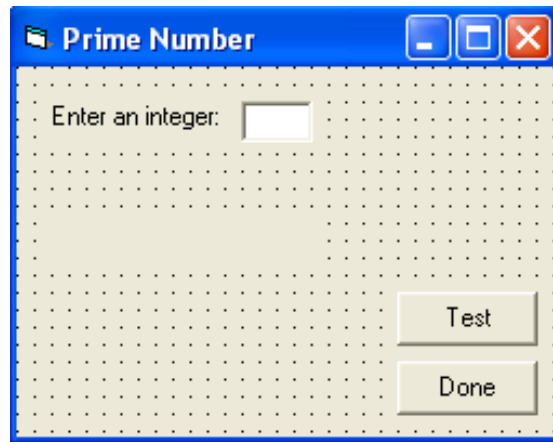
```
    intTestNum = txtInteger.Text  
    intDivisor = 1  
    If intTestNum > 1 Then  
        Do  
            intDivisor = intDivisor + 1  
            Loop While intTestNum Mod intDivisor <> 0  
        End If
```

```
        If intDivisor = intTestNum Then  
            lblPrimeResult.Caption = "Prime number"  
        Else  
            lblPrimeResult.Caption = "Not a prime number"  
        End If  
    End Sub
```

```
Private Sub cmdDone_Click()  
    Unload Me  
End Sub
```

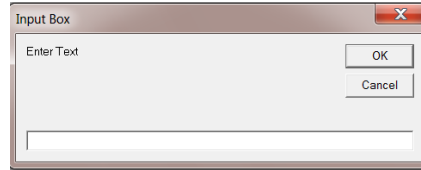
QUESTIONS

1. What is the purpose of the loop in the code?
2. Why did they use the Mod operator in the test of the loop?
3. Is the do... loop while a pretest or posttest loop?
4. Is the do...loop while a fixed or variable loop?



INPUT BOXES

An input box is a dialog box that has a prompt, text box, and OK and Cancel buttons. It is used to get information from the user. Example



Project 2

1. Create the programs below
2. Save it as frmAverageScore
3. Use InputBox to enter data
4. Code the following

Option Explicit

Private intNumberOfScores As Integer, intTotalPoints As Integer

Private Sub cmdEnterScores_Click()

Const strTitle As String = "Grades"

Const strPrompt As String = "Enter a test score (-1 to finish):"

Const intSentinel As Integer = -1

Dim intScore As Integer, strTempScore As String

 lblAverageMessage.Caption = ""

 lblAverage.Caption = ""

 lblNumberOfScores.Caption = ""

 lblScoresMessage.Caption = ""

 intNumberOfScores = 0

 intTotalPoints = 0

 strTempScore = InputBox(strPrompt, strTitle)

If strTempScore = "" **Then**

 intScore = intSentinel

Else

 intScore = strTempScore

End If

Do While intScore <> intSentinel

 intNumberOfScores = intNumberOfScores + 1

 intTotalPoints = intTotalPoints + intScore

 strTempScore = InputBox(strPrompt, strTitle)

If strTempScore = "" **Then**

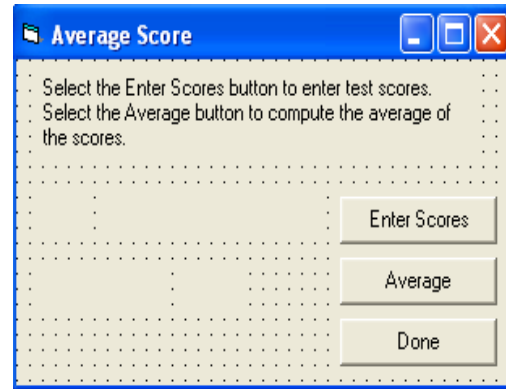
 intScore = intSentinel

Else

 intScore = strTempScore

End If

Loop



' Loop flag

' Clear message

' Clear current average

' Clear number of scores

' Clear scores message

' Initialize global counter

' Initialize global accumulator

' Get score

' Determine if Cancel button was selected in input box

' Test string returned

' Cancel selected

' Score entered

' Count score

' Update total

' Get score

' Determine if Cancel button was selected in input box

' Test string returned

' Cancel selected

' Score entered

```

lblNumberOfScores.Caption = intNumberOfScores
lblScoresMessage.Caption = "scores have been entered"
End Sub

```

```

Private Sub cmdAverage_Click()
    Dim dblAverage As Double
    If intNumberOfScores > 0 Then
        dblAverage = intTotalPoints / intNumberOfScores           ' Compute average
        lblAverageMessage.Caption = "The average is"              ' Display average
        lblAverage.Caption = dblAverage
    Else
        lblAverageMessage.Caption = "The average is"
        lblAverage.Caption = 0
    End If
End Sub

```

```

Private Sub cmdDone_Click()
    Unload Me
End Sub

```

QUESTIONS

1. What is the purpose of intSentinel?
2. What is the purpose of the following statement?
intScore = strTempScore
3. Why was the Input Box statement put in the loop?
4. Why was the statement “dblAverage = intTotalPoints / intNumberOfScores” put in the if statement below?

```

If intNumberOfScores > 0 Then

```

Project 3

1. Create the program below
2. Save it as frmUniqueRandomNumber

Option Explicit

```

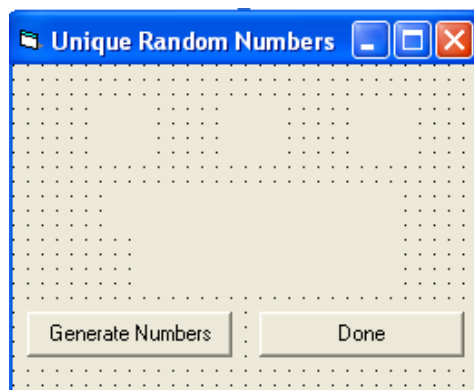
Private Sub Form_Load()
    Randomize
End Sub

```

```

Private Sub cmdGenerateNumbers_Click()
    Dim intMaxNum As Integer
    Dim intNumber1 As Integer, intNumber2 As Integer,
    intNumber3 As Integer
    Dim intNumberOfLoops As Integer

```



Integer,

```

intNumberOfLoops = 0
intMaxNum = InputBox("Enter max number:", "Max Number")
Do
    intNumberOfLoops = intNumberOfLoops + 1
    intNumber1 = Int(intMaxNum * Rnd + 1)
    intNumber2 = Int(intMaxNum * Rnd + 1)
    intNumber3 = Int(intMaxNum * Rnd + 1)
Loop While intNumber1 = intNumber2 Or intNumber2 = intNumber3 Or intNumber1 = intNumber3
lblNumber1.Caption = intNumber1
lblNumber2.Caption = intNumber2
lblNumber3.Caption = intNumber3
lblIterations.Caption = intNumberOfLoops
lblIterationsMessage.Caption = "loop iterations were needed to generate three unique numbers."
End Sub

Private Sub cmdDone_Click()
    Unload Me
End Sub

```

QUESTIONS

1. What is the purpose of the Randomize statement in the Form_Load Statement?
2. What is the purpose of the Int() function?
3. What values do Rnd returns?

Project 4

1. Create the programs below
2. Save it as frmFactorial
3. Code the following

Option Explicit

Private Sub

cmdComputeFactorial_Click()

```

Dim intNumber As Integer
Dim lngFactorial As Long
Dim intCount As Integer

```

```

intNumber = txtNumber.Text
lngFactorial = 1

```

```

For intCount = 1 To intNumber

```

```

    lngFactorial = lngFactorial * intCount

```

```

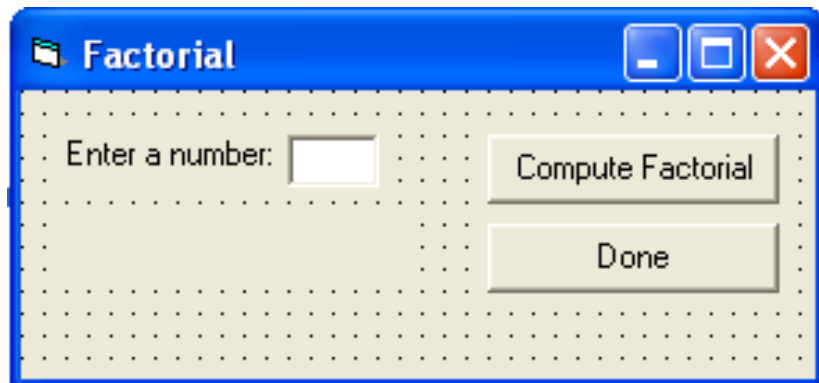
Next intCount

```

```

lblFactorialMessage.Caption = "Factorial is:"

```



```
    lblFactorial.Caption = lngFactorial
```

```
End Sub
```

```
Private Sub Form_Load()
```

```
End Sub
```

```
Private Sub txtNumber_Change()
```

```
    lblFactorialMessage.Caption = ""
```

```
    lblFactorial.Caption = ""
```

```
End Sub
```

```
Private Sub cmdDone_Click()
```

```
    Unload Me
```

```
End Sub
```

QUESTIONS

1. Why did they declare lngFactorial as a Long data type?

2. Is the statement “**lngFactorial = lngFactorial * intCoun**

Project 5

1. Create the programs below
2. Save it as frmOddNumberSum
3. Code the following

Option Explicit

```
Private Sub Form_Load()
```

```
End Sub
```

```
Private Sub txtNumber_Change()
```

```
    lblSumMessage.Caption = ""
```

```
    lblSum.Caption = ""
```

```
End Sub
```

```
Private Sub cmdCalculateSum_Click()
```

```
    Dim intMaxNumber As Integer
```

```
    Dim intSum As Integer
```

```
    Dim intCount As Integer
```

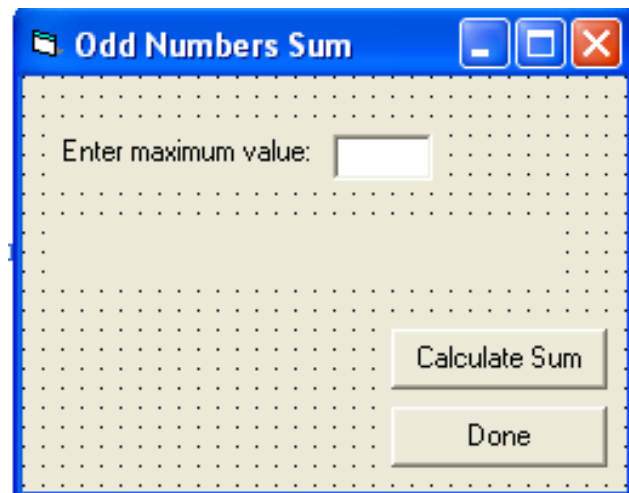
```
    intMaxNumber = txtNumber.Text
```

```
    intSum = 0
```

```
    For intCount = 1 To intMaxNumber Step 2
```

```
        intSum = intSum + intCount
```

```
    Next intCount
```



```

    lblSumMessage.Caption = "The sum of the odd numbers is"
    lblSum.Caption = intSum
End Sub

```

```

Private Sub cmdDone_Click()
    Unload Me
End Sub

```

QUESTION:

1. How did the program sum the odd digits? What was the piece code that causes the next odd to be added?

Project 6

1. Create the programs below
2. Save it as frmSizeIncrement
3. Code the following

Option Explicit

```

Private Sub cmdIncrementFontSize_Click()
    Dim intCounter As Integer
    For intCounter = 10 To 50 Step 5
        lblFontSize.FontSize = intCounter
        MsgBox "Ready to continue?"
    Next
End Sub

```

```

Private Sub cmdDone_Click()
    Unload Me
End Sub

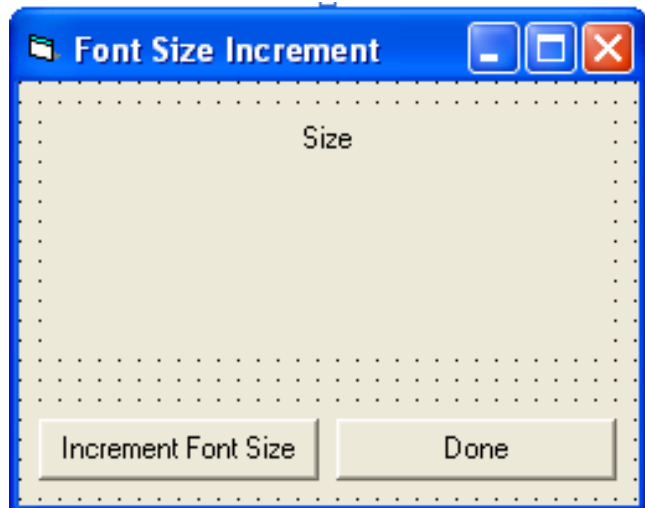
```

```

Private Sub Form_Load()

End Sub

```



Questions

1. What is the purpose of the Step portion of the for loop?
2. When determining which loop to use, what criteria should be use when selecting a loop?
3. Can loops be interchanged?
4. How are loops controlled by users?
5. When working with sentinel values, why must you be careful when using an accumulator?